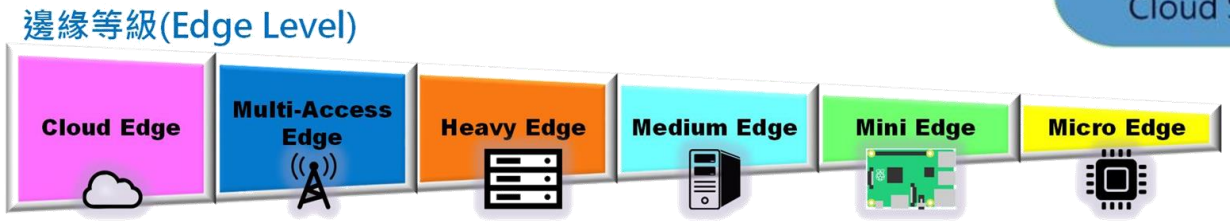
國立臺灣科技大學
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department of Computer Science & Information Engineering
資 訊 工 程 系

人工智慧與邊緣運算實務

7.1 邊緣智慧案例實作【影像分類】



資訊工程系　許哲豪 助理教授

7.1 影像分類

- 範例說明
- 程式碼說明
- 輸出結果展示

DevCloud 範例說明

- 利用Intel DevCloud執行OpenVINO
- 運行DevCloud基本影像分類範例
  tutorial_classification_sample.inpynb
- 本範例以**squeezenet1.1**及對應Label進行展示，並備有dog, cat, bird三張影像供測試用。
- 主要工作流程
  - 載入硬體插件 (Load the plugin)
  - 讀入模型、參數 (Read the model IR)
  - 載入模型到硬體插件 (Load the model into the plugin)
  - 準備輸入影像 (Prepare the input)
  - 執行推論 (Run inference)
  - 處理輸出 (Process the output)

工作流程 – 導入函式庫

- 導入必要函式庫

```
import os
import cv2
import time
import numpy as np
from openvino.inference_engine import IECore
%matplotlib inline
from matplotlib import pyplot as plt
print('Imported Python modules successfully.')
```

```
Imported Python modules successfully.
```

工作流程 – 準備模型、參數

```
!downloader.py --name squeezenet1.1 -o raw_model
```

下載模型

```
################|| Downloading squeezenet1.1 ||################

========== Downloading raw_model/public/squeezenet1.1/squeezenet1.1.prototxt
... 100%, 9 KB, 24203 KB/s, 0 seconds passed

========== Downloading raw_model/public/squeezenet1.1/squeezenet1.1.caffemodel
... 100%, 4834 KB, 17452 KB/s, 0 seconds passed

========== Replacing text in raw_model/public/squeezenet1.1/squeezenet1.1.prototxt
```

```
!converter.py --name squeezenet1.1 -d raw_model -o model
```

轉換模型

```
========== Converting squeezenet1.1 to IR (FP16)
Conversion command: /usr/bin/python3 -- /opt/intel/openvino/deployment_tools/model_op
c/squeezenet1.1/FP16 --model_name=squeezenet1.1 '--input_shape=[1,3,227,227]' --input
=raw_model/public/squeezenet1.1/squeezenet1.1.caffemodel --input_proto=raw_model/publ
```

工作流程 – 配置變數名稱

```
# model IR files
model_xml = "model/public/squeezenet1.1/FP32/squeezenet1.1.xml"
model_bin = "model/public/squeezenet1.1/FP32/squeezenet1.1.bin"

# input image file
input_path = "./dog.jpg"

# CPU extension library to use
cpu_extension_path = os.path.expanduser("~")+"/inference_engine_samples/intel64/Release/lib/libcpu_extension.so"

# device to use
device = "CPU"

# number of top results to display
report_top_n = 10

# output labels
labels_path = "squeezenet1.1.labels"

print("Configuration parameters settings:"
      "\n\tmodel_xml=", model_xml,
       "\n\tmodel_bin=", model_bin,
       "\n\tinput_path=", input_path,
       "\n\tdevice=", device,
       "\n\tlabels_path=", labels_path,
       "\n\treport_top_n=", report_top_n)
```

模型及參數檔名

輸入影像檔名

指定硬體，DevCloud上預設為Xeon所以只能跑CPU，若將程式移到本地端則可運行在GPU和VPU (MYRIAD, HDDL)

輸出置信度最高項次數量

輸出標籤檔

工作流程 – 建立推論引擎

```
# create Inference Engine instance
ie = IECore()
print("An Inference Engine object has been created")
```

建立推論引擎

```
# load network from IR files
net = ie.read_network(model=model_xml, weights=model_bin)
print("Loaded model IR files [",model_bin,"] and [", model_xml, "]\n")

# check to make sure that the plugin has support for all layers in the loaded model
#TODO:replace deprecated layers.keys() with ngraph iterator
'''supported_layers = ie.query_network(net,device)
not_supported_layers = [l for l in net.layers.keys() if l not in supported_layers]
if len(not_supported_layers) != 0:
    print("ERROR: Following layers are not supported by the plugin for specified",
          " device {}:\n {}".format(device, ', '.join(not_supported_layers)))
    assert 0 == 1, "ERROR: Missing support for all layers in the model," \
              + " cannot continue."'''

# check to make sue that the model's input and output are what is expected
assert len(net.input_info.keys()) == 1, \
    "ERROR: This sample supports only single input topologies"
assert len(net.outputs) == 1, \
    "ERROR: This sample supports only single output topologies"
print("SUCCESS: Model IR files have been loaded and verified")
```

讀入網路及參數

檢查模型正確性

工作流程 – 載入模型

載入網路到硬體

```
exec_net = ie.load_network(network=net, num_requests=2, device_name=device)

# store name of input and output blobs
input_blob = next(iter(net.input_info))
output_blob = next(iter(net.outputs))

# read the input's dimensions: n=batch size, c=number of channels, h=height, w=width
n, c, h, w = net.input_info[input_blob].input_data.shape
print("Loaded model into Inference Engine for device:", device,
      "\nModel input dimensions: n=",n,", c=",c,", h=",h,", w=",w)
```

取得網路各節點資訊

取得輸入基本資訊

```
Loaded model into Inference Engine for device: CPU
Model input dimensions: n= 1 , c= 3 , h= 227 , w= 227
```

工作流程 – 載入標籤

```
labels_map = None
# if labels points to a label mapping file, then load the file into labels_map
print(labels_path)
if os.path.isfile(labels_path):
    with open(labels_path, 'r') as f:
        labels_map = [x.split(sep=' ', maxsplit=1)[-1].strip() for x in f]
    print("Loaded label mapping file [",labels_path,"]")
else:
    print("No label mapping file has been loaded, only numbers will be used",
          " for detected object labels")
```

將標籤依序放入labels_map陣列中

工作流程 – 載入影像函數

```
# define function to load an input image
def loadInputImage(input_path):
    # globals to store input width and height
    global input_w, input_h

    # use OpenCV to load the input image
    cap = cv2.VideoCapture(input_path)

    # store input width and height
    input_w = cap.get(3)
    input_h = cap.get(4)
    print("Loaded input image [",input_path,"], resolution=", input_w, "w x ",
          input_h, "h")

    # load the input image
    ret, image = cap.read()
    del cap
    return image
```

開啟影像檔

取得影像尺寸

回傳取得的影像

工作流程 – 影像尺寸縮放

```
# define function for resizing input image
def resizeInputImage(image):
    # resize image dimensions form image to model's input w x h
    in_frame = cv2.resize(image, (w, h))
    # Change data layout from HWC to CHW
    in_frame = in_frame.transpose((2, 0, 1))
    # reshape to input dimensions
    in_frame = in_frame.reshape((n, c, h, w))
    print("Resized input image from {} to {}".format(image.shape[:-1], (h, w)))
    return in_frame
```

影像縮放到指定尺寸

轉置影像陣列從HWC變CHW

將輸入維度重新映射

工作流程 – 載入影像並調整尺寸

```
# load image
image = loadInputImage(input_path)

# resize the input image
in_frame = resizeInputImage(image)

# display input image
print("Input image:")
plt.axis("off")
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
```

讀入測試影像

影像縮放到指定尺寸

顯示測試影像

```
Loaded input image [ ./dog.jpg ], resolution= 3124.0 w x  4020.0 h
Resized input image from (4020, 3124) to (227, 227)
Input image:
<matplotlib.image.AxesImage at 0x7f23d5ae2b38>
```

工作流程 – 執行推論

```
# save start time
inf_start = time.time()

# run inference
res = exec_net.infer(inputs={input_blob: in_frame})

# calculate time from start until now
inf_time = time.time() - inf_start
print("Inference complete, run time: {:.3f} ms".format(inf_time * 1000))
```

開始計時

執行推論

結束計時並計算時長

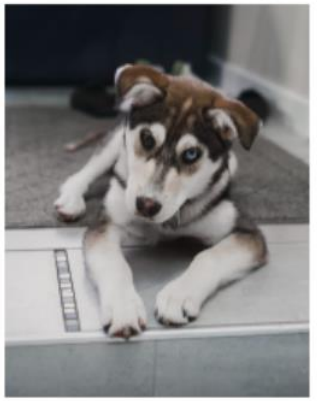
```
Inference complete, run time: 15.451 ms
```

工作流程 – 處理和顯示結果

```python
# create function to process inference results
def processAndDisplayResults(probs, orig_input_image, orig_input_path):
    # display image
    plt.figure()
    plt.axis("off")
    im_to_show = cv2.cvtColor(orig_input_image, cv2.COLOR_BGR2RGB)

    # report top n results for image
    print("Top ", report_top_n, " results for image",orig_input_path,":")

    # remove dimensions of length=1
    probs = np.squeeze(probs)

    # sort then return top report_top_n entries
    top_ind = np.argsort(probs)[-report_top_n:][::-1]

    # show input image
    plt.imshow(im_to_show)

    # print out top probabilities, looking up label
    print("Probability% is <label>")
    for id in top_ind:
        det_label = labels_map[id] if labels_map else "#{}".format(id)
        print(" {:.7f} % is {}".format(probs[id], det_label))
    print("\n")

processAndDisplayResults(res[output_blob][0], image, input_path)
print("Processed and displayed inference output results.")
```

執行結果

```
Top  10  results for image ./dog.jpg :
Probability% is <label>
 0.2329701 % is Chihuahua
 0.0853880 % is husky
 0.0783308 % is dog, husky
 0.0472347 % is pug-dog
 0.0447004 % is malemute, Alaskan malamute
 0.0416504 % is Cardigan Welsh corgi
 0.0400899 % is bull, Boston terrier
 0.0384709 % is collie
 0.0338483 % is ferret, ferret, Mustela nigripes
 0.0301934 % is terrier

Processed and displayed inference output results.
```



工作流程 – 推論其它影像

```python
# define function to prepare input, run inference, and process inference results
def inferImage(image, input_path):
    # prepare input
    in_frame = resizeInputImage(image)

    # run inference
    res = exec_net.infer(inputs={input_blob: in_frame})

    # process inference results
    processAndDisplayResults(res[output_blob][0], image, input_path)

# set path to different input image
input_path="cat.jpg"

# load input image
image = loadInputImage(input_path)

# infer image and display results
inferImage(image, input_path)
```

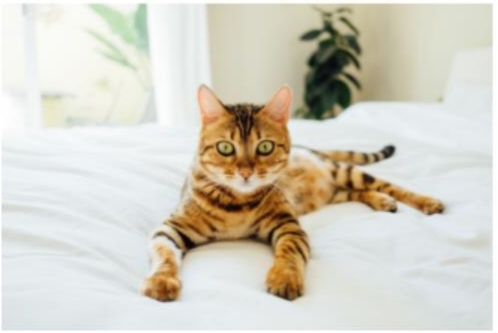
指定新影像

亦可指定網路圖檔

```python
# input_path may be set to a local file or URL
input_path = "https://cdn.pixabay.com/photo/2015/06/25/12/27/daisy-821222_1280.jpg"
```

執行結果

```
Loaded input image [ cat.jpg ], resolution= 4482.0 w x  2988.0 h
Resized input image from (2988, 4482) to (227, 227)
Top  10  results for image cat.jpg :
Probability% is <label>
 0.5989718 % is tabby cat
 0.2263101 % is cat
 0.0743189 % is catamount
 0.0419797 % is cat
 0.0258655 % is Panthera tigris
 0.0116104 % is bathing tub, bath, tub
 0.0063771 % is vat
 0.0018472 % is terrier
 0.0015122 % is terrier
 0.0013558 % is terrier
```



工作流程 – 載入批次輸入影像

```python
# define function to load input images into input batch
def batchLoadInputImages(batch_paths):
    global batch_size
    global batch_images
    global orig_image_paths
    global orig_images
    batch_size = len(batch_paths)

    # create input batch (array) of input images
    batch_images = np.ndarray(shape=(batch_size, c, h, w))

    # create array to hold original images and paths for displaying later
    orig_images = []
    orig_image_paths = []

    for i in range(batch_size):
        # load image
        image = loadInputImage(batch_paths[i])

        # save original image and path
        orig_images.append(image)
        orig_image_paths.append(batch_paths[i])

        # prepare input
        in_frame = resizeInputImage(image)

        # add input to batch
        batch_images[i] = in_frame
    return batch_size, batch_images, orig_image_paths, orig_images
```

把影像放入陣列

```python
# batch of inputs which may be local files or URLs (comma separated)
batch_paths = ["./dog.jpg", "./cat.jpg", "./bird.jpg"]
```

```python
batchLoadInputImages(batch_paths)
print("Loaded", batch_size, "images.")
```

```
Loaded input image [ ./dog.jpg ], resolution= 3124.0 w x  4020.0 h
Resized input image from (4020, 3124) to (227, 227)
Loaded input image [ ./cat.jpg ], resolution= 4482.0 w x  2988.0 h
Resized input image from (2988, 4482) to (227, 227)
Loaded input image [ ./bird.jpg ], resolution= 3264.0 w x  2448.0 h
Resized input image from (2448, 3264) to (227, 227)
Loaded 3 images.
```

工作流程 – 執行批次推論

```
# set the batch size to match the number of input images
net.batch_size = batch_size
print("Network batch size set to", batch_size)

## reload network because batch size has changed
exec_net = ie.load_network(network=net, num_requests=2, device_name=device)


# save start time
inf_start = time.time()

# run inference
res = exec_net.infer(inputs={input_blob: batch_images})

# calculate time from start until now
inf_time = time.time() - inf_start
print("Inference complete, run time: {:.3f} ms".format(inf_time * 1000))
```

取得批次影像數量

重新載入網路

推論批次影像

```
Network batch size set to 3
Inference complete, run time: 22.151 ms
```

工作流程 – 批次處理和顯示輸出

```
# create function to process inference results
def batchProcessAndDisplayResults(result, orig_input_images, orig_image_paths):
    # get output results
    res = result[output_blob]

    for i, probs in enumerate(res):
        processAndDisplayResults(probs, orig_input_images[i], orig_image_paths[i])

# process inference results
batchProcessAndDisplayResults(res, orig_images, orig_image_paths)
```

```
Top  10  results for image ./dog.jpg :
Probability% is <label>
 0.2329701 % is Chihuahua
 0.0853880 % is husky
 0.0783308 % is dog, husky
 0.0472347 % is pug-dog
 0.0447004 % is malemute, Alaskan malamute
 0.0416504 % is Cardigan Welsh corgi
 0.0400899 % is bull, Boston terrier
 0.0384709 % is collie
 0.0338483 % is ferret, ferret, Mustela nigripes
 0.0301934 % is terrier
```

```
Top  10  results for image ./cat.jpg :
Probability% is <label>
 0.5989718 % is tabby cat
 0.2263101 % is cat
 0.0743189 % is catamount
 0.0419797 % is cat
 0.0258655 % is Panthera tigris
 0.0116104 % is bathing tub, bath, tub
 0.0063771 % is vat
 0.0018472 % is terrier
 0.0015122 % is terrier
 0.0013558 % is terrier
```

```
Top  10  results for image ./bird.jpg :
Probability% is <label>
 0.9974803 % is magpie
 0.0014055 % is oyster catcher
 0.0003362 % is ouzel, dipper
 0.0002929 % is mollymawk
 0.0002199 % is stork, Ciconia nigra
 0.0000741 % is goose
 0.0000670 % is drake
 0.0000194 % is hornbill
 0.0000120 % is turnstone, Arenaria interpres
 0.0000109 % is merganser, Mergus serrator
```

工作流程 – 顯示網路結構

```
# retrieve performance counters from last inference request
perf_counts = exec_net.requests[0].get_perf_counts()

# display performance counters for each layer
print("Performance counters:")
print("{:<40} {:<15} {:<15} {:<15} {:<10}".format('name', 'layer_type',
        'exec_type', 'status', 'real_time, us'))
for layer, stats in perf_counts.items():
    print("{:<40} {:<15} {:<15} {:<15} {:<10}".format(layer,
        stats['layer_type'], stats['exec_type'],
        stats['status'], stats['real_time']))
```

```
Performance counters:
name                                     layer_type      exec_type       status          real_time, us
conv1                                    Convolution     jit_avx512_FP32 EXECUTED        3260
conv10                                   Convolution     jit_avx512_1x1_FP32 EXECUTED        2577
data/mean                                ScaleShift      jit_avx512_FP32 EXECUTED        729
data/mean_const_biases                   Const           unknown_FP32    NOT_RUN         0
data/mean_const_weights                  Const           unknown_FP32    NOT_RUN         0
fire2/concat                             Concat          unknown_FP32    EXECUTED        2
fire2/expand1x1                          Convolution     jit_avx512_1x1_FP32 EXECUTED        209
fire2/expand3x3                          Convolution     jit_avx512_FP32 EXECUTED        1356
fire2/relu_expand1x1                     ReLU            undef           NOT_RUN         0
fire2/relu_expand3x3                     ReLU            undef           NOT_RUN         0
fire2/relu_squeeze1x1                    ReLU            undef           NOT_RUN         0
fire2/squeeze1x1                         Convolution     jit_avx512_1x1_FP32 EXECUTED        213
fire3/concat                             Concat          unknown_FP32    EXECUTED        2
```

網路結構輸出結果

Google Colab 範例說明

- 使用OpenVINO預訓練模型**resnet-34-pytorch**進行影像分類

*** 原始Github範例**

https://github.com/OmniXRI/Colab_DevCloud_OpenVINO_Samples/blob/main/Colab_OpenVINO_Image_Classification.ipynb

*** 直接使用Colab啟動Github中ipynb程式**

https://colab.research.google.com/github/OmniXRI/Colab_DevCloud_OpenVINO_Samples/blob/main/Colab_OpenVINO_Image_Classification.ipynb

程式碼說明 (1/6)

- 1. 安裝Intel OpenVINO工具包
  - 以apt方式安裝OpenVINO，安裝版本為2021.3.394預設安裝路徑為/opt/intel/OpenVINO_2021.3.394，系統會自建出/opt/intel/OpenVINO_2021捷徑名稱，後續可使用這個較短捷徑名稱。

```
# 顯示目前工作目錄
!pwd
# 取得OpenVINO 2021公開金鑰
!wget https://apt.repos.intel.com/openvino/2021/GPG-PUB-KEY-INTEL-OPENVINO-2021
# 加入OpenVINO公開金鑰到系統金鑰群中
!apt-key add GPG-PUB-KEY-INTEL-OPENVINO-2021
# 建立更新安裝清單檔案
!touch /etc/apt/sources.list.d/intel-openvino-2021.list
# 將下載指令加入安裝清單中
!echo "deb https://apt.repos.intel.com/openvino/2021 all main" >> /etc/apt/sources.list.d/intel-openvino-2021.list
# 更新系統
!apt update
# 安裝OpenVINO到虛擬機系統中
!apt install intel-openvino-dev-ubuntu18-2021.3.394
# 列出安裝路徑下內容進行確認
!ls /opt/intel
```

程式碼說明 (2/6)

- 2. 下載模型
  - OpenVINO Open Model Zoo (Public pretrained models)中提供了約有70多種現成的影像分類模型，如下列網址所示。
    https://docs.openvinotoolkit.org/latest/classification_models_public.html
  - 這裡選用 **--name resnet-34-pytorch** （可自行變更所需模型名稱）

```
!source /opt/intel/openvino_2021/bin/setupvars.sh && \
python3 /opt/intel/openvino_2021/deployment_tools/tools/model_downloader/downloader.py --name resnet-34-pytorch
```

程式碼說明 (3/6)

- 3.模型轉換
  - 如果下載的是Intel Pretrained Model則不需轉換就自帶IR檔(xml,bin)若是Public Pretrained Model則須進行轉換成IR檔，系統會自動判別。--name 參數為待轉換模型名稱

```
# 因為轉換PyTorch會用到ONNX，所以要安裝相關套件包
!pip3 install onnx

# 下載及安裝test-generator 方便檢查程式運行錯誤
!pip3 install test-generator==0.1.1

# 執行環境設定批次檔並將下載到的模型檔進行轉換產生IR(xml & bin)檔
!source /opt/intel/openvino_2021/bin/setupvars.sh && \
python3 /opt/intel/openvino_2021/deployment_tools/tools/model_downloader/converter.py \
--name resnet-34-pytorch

# 檢查模型轉檔後會產生/FP16, FP32不同精度的IR檔(xml, bin)
!ls public/resnet-34-pytorch
!ls public/resnet-34-pytorch/FP32
```

程式碼說明 (4/6)

- 4. 準備測試影像
  - 從網路獲取任意一張測試影像並顯示

```
# 以OpenCV檢視輸入影像
import cv2
import matplotlib.pyplot as plt
import numpy as np
import requests

# 從網路獲取一張影像
file = requests.get("https://images.chinatimes.com/newsphoto/2021-04-05/1024/20210405002742.jpg")
# 將影像轉成OpenCV格式存入img中
img = cv2.imdecode(np.frombuffer(file.content, np.uint8), 1)
# 將img寫入磁碟命名為input.jpg
cv2.imwrite('input.jpg',img)

# 亦可直接讀取本地端影像
# img = cv2.imread('input.jpg')

rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # 將影像從BGR格式轉到RGB格式，才能讓plt.imshow()正確顯示
plt.figure() # 準備一顯示影像空間
plt.axis("off") # 設定關閉XY軸刻尺
plt.imshow(rgb) # 顯示影像
```

程式碼說明 (5/6)

- 5.進行推論
  - 在 OpenVINO 中大部份的範例程式都放在/opt/intel/openvino_2021/deployment_tools/inference_engine/demos中，但影像辨識是最基本的範例，所以不在其中，而是放在 /opt/intel/openvino_2021/inference_engine/samples/python/下。這裡使用/python/hello_classification/hello_classification.py來進行影像分類測試。這個範例適用上述數十種模型。另外要依據不同的模型所對應的資料集給予標籤檔案，預設放在/opt/intel/openvino_2021/deployment_tools/open_model_zoo/data/dataset_classes下。
  - 輸入參數：
    - -i 輸入影像名稱 (.png, .jpg ...)
    - -m 模型名稱 (.xml)
    - -nt 輸出置信度排序最高的前幾項（選配參數，預設為Top 10）
    - --labels 標籤名稱檔(.txt) （選配參數，不使用時會以id編號輸出，這裡要搭配無id之標籤檔案）
  - 最後顯示推論結果，包括分類編號(classid)及置信度(probability)

程式碼說明 (6/6)

```
# 列出可支援標籤檔案(此步驟可忽略)
!ls /opt/intel/openvino_2021.3.394/deployment_tools/open_model_zoo/data/dataset_classes
```

```
aclnet_53cl.txt       coco_91cl.txt          msasl100.json
cityscapes_19cl.txt   imagenet_2012.txt      scut_ept.txt
coco_80cl_bkgr.txt    imagenet_2015.txt      voc_20cl_bkgr.txt
coco_80cl.txt         jester27.json          voc_20cl.txt
coco_91cl_bkgr.txt    kondate_nakayosi.txt
```

執行推論

```
# 設定環境變數執行影像分類推論
!source /opt/intel/openvino_2021/bin/setupvars.sh && \
python3 \
/opt/intel/openvino_2021/inference_engine/samples/python/hello_classification/hello_classification.py \
-i 'input.jpg' \
-m public/resnet-34-pytorch/FP32/resnet-34-pytorch.xml \
--labels "imagenet_label.txt"
```

輸入影像名稱
模型名稱
標籤檔名稱

輸出結果展示

```
GPG-PUB-KEY-INTEL-OPENVINO-2021  input.jpg  sample_data
imagenet_label.txt               public
[setupvars.sh] OpenVINO environment initialized
[ INFO ] Creating Inference Engine
[ INFO ] Loading network:
        public/resnet-34-pytorch/FP32/resnet-34-pytorch.xml
[ INFO ] Preparing input blobs
[ WARNING ] Image input.jpg is resized from (576, 1024) to (224, 224)
[ INFO ] Loading model to the plugin
[ INFO ] Starting inference in synchronous mode
[ INFO ] Processing output blob
[ INFO ] Top 10 results:
Image input.jpg

classid probability
------- -----------
dog, husky 20.7688637
 husky    19.8722458
malemute, Alaskan malamute17.8152275
elkhound, elkhound12.2419424
collie    11.7569017
wolf, grey wolf, gray wolf, Canis lupus10.4696951
Cardigan Welsh corgi10.2310362
dog sled, dog sleigh10.2000027
shepherd, German shepherd dog, German police dog, alsatian10.0990477
collie    9.7165003
```

參考文獻

- NTUST Edge AI ChC Intel DevCloud安裝與測試

https://omnixri.blogspot.com/p/ntust-edge-ai-chc.html

- Intel DevCloud

https://devcloud.intel.com/edge/

- Github – OmniXRI – Colab_DevCloud_OpenVINO_Samples

https://github.com/OmniXRI/Colab_DevCloud_OpenVINO_Samples